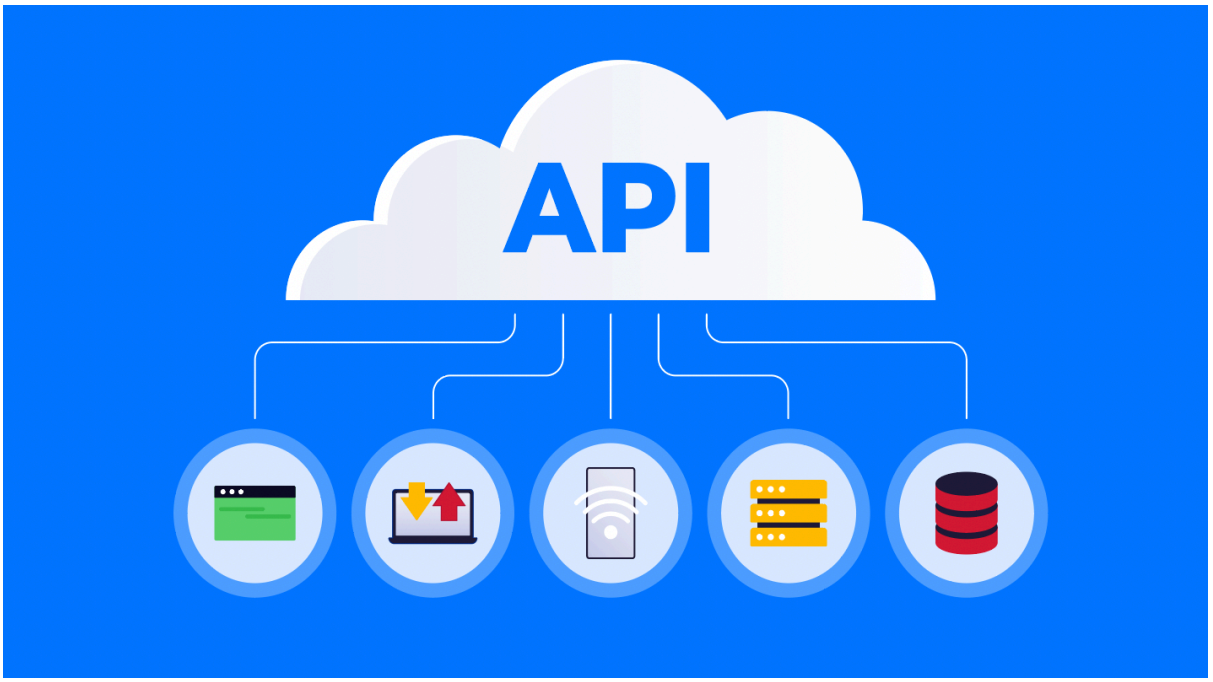
Proyecto UT7 – Servicios y uso de APIs



Ismael Díaz Morales

ÍNDICE:

Ejercicio 1:

Servicio.php:

Primero se crea la clase ServiciosSOAP, esta clase define los métodos que el servicio SOAP ofrecerá a los clientes.

Dentro de aquí está la conexión a la base de datos.

El método infoMódulo que obtiene la información de un módulo.

```php
public function infoModulo(int $id)
{
    try {
        $stmt = $this->conexion->prepare("SELECT * FROM modulos WHERE id = :id");
        $stmt->execute(["id" => $id]);
        $resultado = $stmt->fetch(PDO::FETCH_ASSOC);

        return json_encode($resultado ?: ["error" => "Módulo no encontrado"]);
    } catch (PDOException $e) {
        return json_encode(["error" => "Error en la consulta de módulo: " . $e->getMessage()]);
    }
}
```

El método infoDepartamentos que obtiene los departamentos disponibles:

```php
public function infoDepartamentos()
{
    try {
        $stmt = $this->conexion->query("SELECT DISTINCT Departamento FROM modulos");
        $resultados = $stmt->fetchAll(PDO::FETCH_COLUMN);

        return json_encode($resultados ?: ["error" => "No hay departamentos disponibles"]);
    } catch (PDOException $e) {
        return json_encode(["error" => "Error en la consulta de departamentos: " . $e->getMessage()]);
    }
}
```

El método infoNomenclaturas que obtiene las nomenclaturas.

```php
public function infoNomenclaturas()
{
    try {
        $stmt = $this->conexion->query("SELECT DISTINCT nomenclatura FROM modulos");
        $resultados = $stmt->fetchAll(PDO::FETCH_COLUMN);

        return json_encode($resultados ?: ["error" => "No hay nomenclaturas disponibles"]);
    } catch (PDOException $e) {
        return json_encode(["error" => "Error en la consulta de nomenclaturas: " . $e->getMessage()]);
    }
}
```

Y fuera de la clase ServiciosSOAP se configura el servidor SOAP

```php
// Configuración del servidor SOAP
$options = [
    'uri' => 'http://localhost/SOAP/',
    'soap_version' => SOAP_1_2
];

$server = new SoapServer(null, $options);
$server->setClass('ServiciosSOAP');
$server->handle();
?>
```

Cliente.php
Llamadas a los Métodos del Servicio SOAP

```php
try {
    // Obtener módulo por ID
    $modulo = json_decode($cliente->infoModulo(1), true);

    // Obtener departamentos
    $departamentos = json_decode($cliente->infoDepartamentos(), true);

    // Obtener nomenclaturas
    $nomenclaturas = json_decode($cliente->infoNomenclaturas(), true);

} catch (SoapFault $e) {
    $error = "Error en el servicio SOAP: " . $e->getMessage();
} catch (Exception $e) {
    $error = "Error general: " . $e->getMessage();
}
?>
```

Se hacen tres llamadas al servidor SOAP para obtener:
Información de un módulo con ID 1.
Lista de departamentos.
Lista de nomenclaturas.
Cada respuesta viene en formato JSON, por lo que se usa json_decode(..., true) para convertirla en un array asociativo en PHP.
Si hay un error en la comunicación SOAP (SoapFault), se captura y muestra un mensaje de error.
También se captura cualquier otro tipo de error (Exception).

Aquí se muestra toda la información:

```php
<div class="container">
    <?php if (isset($error)): ?>
        <p class="error"><?php echo $error; ?></p>
    <?php else: ?>
        <h2>Módulo</h2>
        <?php echo $modulo ? "<pre>" . print_r($modulo, true) . "</pre>" : "<p>No existen datos en el módulo</p>"; ?>

        <h2>Departamentos</h2>
        <?php echo $departamentos ? "<pre>" . print_r($departamentos, true) . "</pre>" : "<p>No existen datos en los departamentos</p>"; ?>

        <h2>Nomenclaturas</h2>
        <?php echo $nomenclaturas ? "<pre>" . print_r($nomenclaturas, true) . "</pre>" : "<p>No existen datos en las nomenclaturas</p>"; ?>
    <?php endif; ?>
</div>
```

Ejercicio 2:

Primero cargamos el RSS de EuropaPress:

```php
<?php
// URL del RSS de EuropaPress
$rss_url = "https://www.europapress.es/rss/rss.aspx?ch=00066";

// Cargar XML del RSS
$rss = simplexml_load_file($rss_url);
if ($rss === false) {
    die("Error al cargar el canal RSS.");
}
?>
```

Se define la URL del feed RSS de EuropaPress.
Se usa simplexml_load_file() para cargar el contenido del RSS y convertirlo en un objeto de tipo SimpleXMLElement.
Si hay un error al cargar el RSS, se detiene la ejecución con die() y se muestra un mensaje de error.

```php
<?php foreach ($rss->channel->item as $item): ?>
    <tr>
        <td><?php echo $item->title; ?></td>
        <td><?php echo $item->description; ?></td>
        <td><a href="<?php echo $item->link; ?>" target="_blank">Ver noticia</a></td>
    </tr>
<?php endforeach; ?>
```

Se usa foreach para recorrer cada elemento <item> dentro del canal RSS.
Se imprimen:
Título de la noticia ($item->title).
Descripción ($item->description).
Enlace ($item->link), con un botón que dice "Ver noticia".
La etiqueta target="_blank" hace que el enlace se abra en una nueva pestaña.

Ejercicio 3:

Primero tenemos datos.js donde tenemos la API de aemet.
Tenemos una función llamada obtenerMapaIsobaras:

```
async function obtenerMapaIsobaras() {
  try {
    const response = await fetch(`https://opendata.aemet.es/opendata/api/mapasygraficos/analisis?api_key=${API_KEY}`)
    const data = await response.json()
    const blobResponse = await fetch(data.datos)
    const blob = await blobResponse.blob()
    const imageUrl = URL.createObjectURL(blob)
    document.getElementById("resultado").innerHTML = `<img src="${imageUrl}" alt="Mapa de isobaras">`
  } catch (error) {
    console.error("Error:", error)
    document.getElementById("resultado").innerHTML = `<p>Error al obtener el mapa de isobaras: ${error.message}</p>`
  }
}
```

Hace una petición a la API de AEMET para obtener el mapa de isobaras.
Convierte la respuesta a JSON.
La API devuelve una URL (data.datos) con la imagen del mapa.
Descarga la imagen en blob (un formato binario).
Crea una URL temporal con URL.createObjectURL(blob).
Muestra la imagen en la página.

Tenemos otra función llamada obtenerInfoCanarias:

```
async function obtenerInfoCanarias() {
  try {
    const response = await fetch(`https://opendata.aemet.es/opendata/api/prediccion/ccaa/hoy/coo/?api_key=${API_KEY}`)
    const data = await response.json()
    const bufferResponse = await fetch(data.datos)
    const buffer = await bufferResponse.arrayBuffer()
    const decoder = new TextDecoder("iso-8859-1")
    const text = decoder.decode(buffer)
    let prediccion
    try {
      const data = JSON.parse(text)
      prediccion = data[0].prediccion
    } catch (e) {
      console.warn("La respuesta no es JSON válido. Mostrando texto plano.")
      prediccion = text
    }
    document.getElementById("resultado").innerHTML = `
            <table>
                <tr><th>Predicción Canarias</th></tr>
                <tr><td>${prediccion}</td></tr>
            </table>
        `
  } catch (error) {
    console.error("Error:", error)
    document.getElementById("resultado").innerHTML =
      `<p>Error al obtener la información de Canarias: ${error.message}</p>`
  }
}
```

Llama a la API de AEMET para obtener la predicción meteorológica de Canarias.
Convierte la respuesta en JSON.
La API devuelve otra URL (data.datos) con la información.
Descarga los datos en un formato binario (arrayBuffer()).
Decodifica la información usando TextDecoder("iso-8859-1") (porque AEMET usa este formato en lugar de UTF-8).
Intenta convertirlo en JSON (JSON.parse(text)).
Si es válido, extrae la predicción.
Si no es JSON válido, muestra el texto sin procesar.

Muestra la predicción en una tabla.

Tenemos otra llamada obtenerInfoGranCanaria:

```
async function obtenerInfoGranCanaria() {
    try {
      const response = await fetch(`https://opendata.aemet.es/opendata/api/prediccion/provincia/manana/353/?api_key=${API_KEY}`);
      const data = await response.json();

      // Hacer una segunda solicitud a la URL contenida en `datos`
      const dataUrl = data.datos;
      const bufferResponse = await fetch(dataUrl);
      const buffer = await bufferResponse.arrayBuffer();

      // Decodificar el contenido (el archivo es probablemente en ISO-8859-1)
      const decoder = new TextDecoder("iso-8859-1");
      const text = decoder.decode(buffer);

      let prediccion;
      try {
        // Intenta parsear el texto como JSON
        const data = JSON.parse(text);
        prediccion = data[0].prediccion;
      } catch (e) {
        // Si no es JSON válido, mostrar el texto plano
        console.warn("La respuesta no es JSON válido. Mostrando texto plano.");
        prediccion = text;
      }

      // Mostrar la predicción
      document.getElementById("resultado").innerHTML = `
        <table>
          <tr><th>Predicción Gran Canaria - Mañana</th></tr>
          <tr><td>${prediccion}</td></tr>
        </table>
      `;
    } catch (error) {
      console.error("Error:", error);
      document.getElementById("resultado").innerHTML =
        `<p>Error al obtener la predicción de Gran Canaria para mañana: ${error.message}</p>`;
    }
  }
```

Llama a la API para obtener la predicción del tiempo en Gran Canaria para mañana.
Convierte la respuesta en JSON.
Descarga los datos en binario y los decodifica.
Intenta convertirlo en JSON.
Si el JSON es válido, muestra la predicción; si no, muestra el texto sin procesar.
Muestra la predicción en una tabla.

Servicio_ahemet.php:
Función getIsobaras

```php
function getIsobaras($api_key, $base_url)
{
    $url = $base_url . "mapasygraficos/analisis?api_key=" . $api_key;
    $response = file_get_contents($url);

    // Verifica si la respuesta fue exitosa
    if ($response === FALSE) {
        die("Error al obtener datos de AEMET.");
    }

    $data = json_decode($response, true);

    if (isset($data["datos"])) {
        return $data["datos"]; // URL de la imagen
    } else {
        die("Error: No se pudieron obtener los datos.");
    }
}
```

Construye la URL de la API para obtener el mapa de isobaras.
Usa file_get_contents($url) para hacer la petición HTTP y obtener la respuesta.
Verifica si hubo un error al hacer la solicitud ($response === FALSE).
Decodifica la respuesta JSON a un array asociativo (json_decode($response, true)).
Si la respuesta contiene "datos", devuelve la URL del mapa de isobaras.
Si no hay "datos", muestra un error.

Función getPrediccionCanarias:

```php
function getPrediccionCanarias($api_key, $base_url)
{
    $url = $base_url . "prediccion/ccaa/hoy/16?api_key=" . $api_key; // Canarias: CCAA ID 16
    $response = file_get_contents($url);

    if ($response === FALSE) {
        die("Error al obtener datos de AEMET.");
    }

    $data = json_decode($response, true);

    if (isset($data["datos"])) {
        return $data["datos"];
    } else {
        die("Error: No se pudieron obtener los datos.");
    }
}
```

Construye la URL de la API para obtener la predicción de Canarias.
Hace una petición con file_get_contents($url).
Verifica errores en la respuesta ($response === FALSE).
Decodifica el JSON en un array asociativo.
Si "datos" existe en la respuesta, lo devuelve.
Si no, muestra un error.

Función getPrediccionGranCanaria:

```php
function getPrediccionGranCanaria($api_key, $base_url)
{
    $url = $base_url . "prediccion/provincia/manana/35?api_key=" . $api_key; // Gran Canaria: Provincia 35
    $response = file_get_contents($url);

    if ($response === FALSE) {
        die("Error al obtener datos de AEMET.");
    }

    $data = json_decode($response, true);

    if (isset($data["datos"])) {
        return $data["datos"];
    } else {
        die("Error: No se pudieron obtener los datos.");
    }
}
```

Construye la URL de la API para obtener la predicción del día de mañana en Gran Canaria.
Hace la petición y obtiene la respuesta.
Verifica errores.
Decodifica el JSON en un array asociativo.
Si "datos" existe en la respuesta, lo devuelve.
Si no, muestra un error.

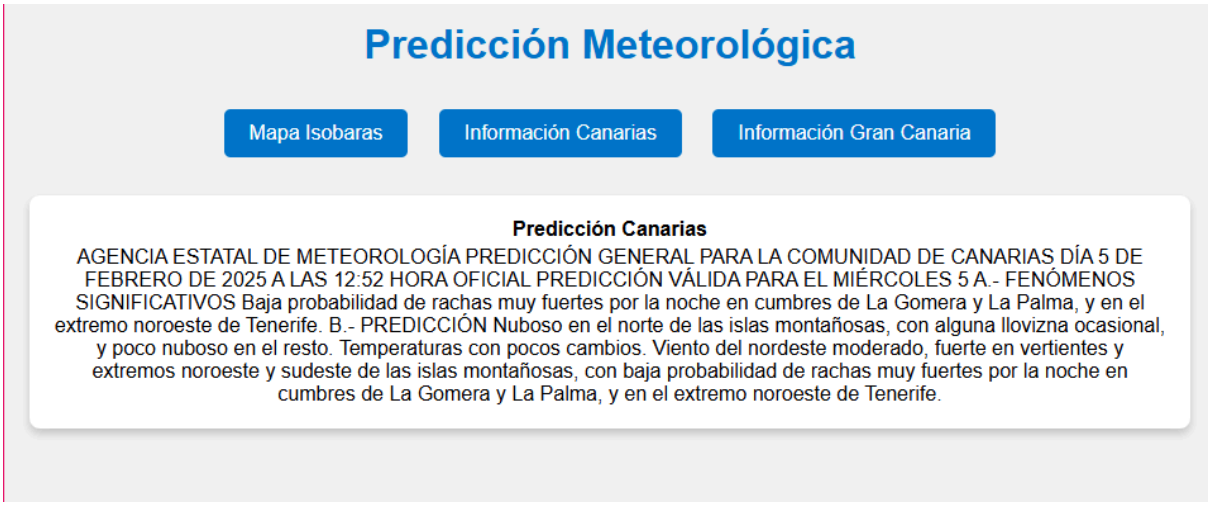
Manejo de solicitudes de frontend:

```php
$accion = isset($_GET['accion']) ? $_GET['accion'] : '';
switch ($accion) {
    case 'isobaras':
        echo getIsobaras($api_key, $base_url);
        break;
    case 'canarias':
        echo getPrediccionCanarias($api_key, $base_url);
        break;
    case 'gran_canaria':
        echo getPrediccionGranCanaria($api_key, $base_url);
        break;

}
```

Obtiene el parámetro acción de la URL ($_GET['accion']).
Evalúa el valor de acción con switch:
Si es "isobaras", llama a getIsobaras() y devuelve la URL del mapa de isobaras.
Si es "canarias", llama a getPrediccionCanarias() y devuelve la predicción de Canarias.
Si es "gran_canaria", llama a getPrediccionGranCanaria() y devuelve la predicción de Gran Canaria.

Y por último tengo el html con los botones para mostrar cada cosa



Predicción Meteorológica

Mapa Isobaras | Información Canarias | Información Gran Canaria

Predicción Gran Canaria - Mañana

AGENCIA ESTATAL DE METEOROLOGÍA PREDICCIÓN PARA LA ISLA DE GRAN CANARIA DÍA 9 DE FEBRERO DE 2025 A LAS 13:01 HORA OFICIAL PREDICCIÓN VÁLIDA PARA EL LUNES 10 GRAN CANARIA Despejado con intervalos en litorales este y norte de madrugada. Calima ligera, más significativa en medianías por la tarde. Temperaturas con pocos cambios. Viento en general flojo del nordeste, predominando las brisas durante las horas centrales. TEMPERATURAS MÍNIMAS Y MÁXIMAS PREVISTAS (°C): Palmas de Gran Canaria, Las 17 21

Predicción Meteorológica

Mapa Isobaras | Información Canarias | Información Gran Canaria